Augustinus-Gymnasium Weiden, Wirtschaftsinformatik 9. JgSt

Grundlagen der Datenbanktechnik

Martin Klier, DBA
09.01.2008

1. Überblick

1.1. Agenda

1.2. Vorstellung Referent

- Martin Klier, 29
- Linux- und Datenbankadministrator, Schwerpunkt hochverfügbare Systeme, Cluster und Replikation
- Arbeitgeber: Klug integrierte Systeme GmbH, Teunz
- Linux seit 1997
- Oracle Database seit 2003
- Kontakt: martin.klier@unix.net
- Web: http://www.usn-it.de

1.3. Do you speak English?

Die meisten Begriffe aus dem IT- und ganz besonders dem Datenbankumfeld entstammen der englischen Sprache, oft gibt es keine aussagekräftigen deutschen Entsprechungen. Dieses Handout liefert für diese Begriffe eine deutsche Erklärung mit, verzichtet aber bewußt nicht auf ihren Einsatz. Englisch ist wie für den Mediziner das Latein die internationale Fachsprache der IT.

2. Einführung

2.1. Was ist eine Datenbank?

„Die wesentliche Aufgabe eines Datenbanksystems ist es, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen.“ (Wikipedia)

a) Strukturierte Ablage

   Bei für den Menschen nicht mehr überschaubaren Datenmengen ist es notwendig, logische Verknüpfungen in der Tradition des Karteikastens (Querverweise) sowie logische Bezüge (Relationen) einzuführen. Auf elektronischem Weg ist es jedoch leichter, mehrdimensionale Verknüpfungen herzustellen als es das in einer manuellen Ablage war. Siehe ER-Diagramm.

b) Schnellüberblick „Nutzung“

   Weiter unten wird das Thema „Anwendung“ noch im Detail behandelt. Vorab soll als Gedankenspielplatz im einfachsten Fall eine normalisierte Adresskartei (z.B. PLZ+Ort in extra Tabelle) und im weiteren Verlauf ein einfacher Webshop/Onlineshop dienen.

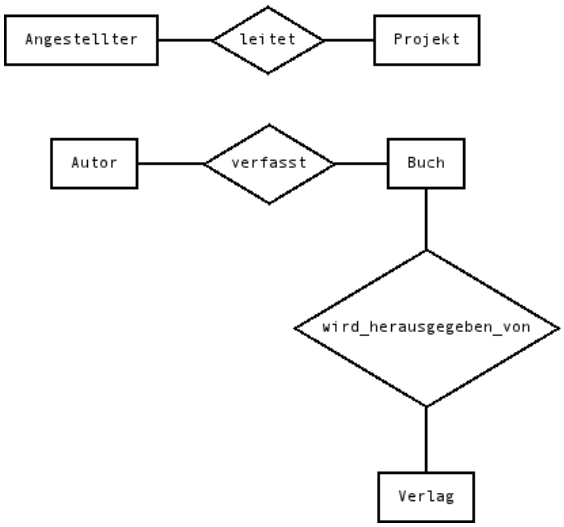
2.2. Geschichte der Relationalen Datenbanksysteme (RDBMS)

- 1970 Theoretische Grundlagen von Edgar F. Codd (IBM Almaden)
- 1974 Ingres RDBMS (QUEL)
- 1977 IBM System R, Sonderanfertigung für Pratt&Whitney (SEQUEL)
- 1977 SDL (Vorläufer von Oracle) erstellt SEQUEL-RDMBS für die CIA
- 1979 RSI (ex SDL) bringt „Oracle RDBMS 2.0“ auf den Markt (SQL)
- 1984 Erstes RDMBS mit Lesekonsistenz
- 1987 RDBM-Systeme für UNIX kommen auf den Markt
- 1988 Prozedurale Programmierung möglich (PL/SQL)
- 1995 64bit-RDBMS erscheinen
- 1997 Datenbanken werden für Internetapplikationen optimiert
- 1999 XML-Unterstützung
- 2003 Datenbank-Server-Cluster mit voller Cachekohärenz
- 2005 Ableger der großen Enterprise-RDBMS werden kostenlos verfügbar

3. Technische Realisierung

3.1. Praxisbeispiel

a) ER-Diagramm



b) Eine einfache Abfrage

```
select a.KD_NR, a.NAME, a.VORNAME
        from TAB_KUNDEN a
        where a.NAME='Schneider';
```

c) Join zweier Tabellen

```
select b.AUFTRAG_NR
        from TAB_KUNDEN a, TAB_AUFTRAEGE b
        where a.KD_NR=b.KD_NR
        and a.KD_NR=1122;
```

d) Ein einfacher Update-Befehl

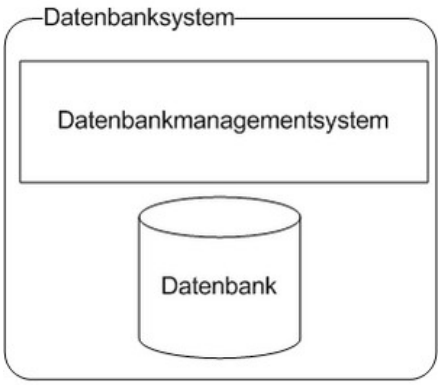
```
update TAB_KUNDEN a
        set a.NAME='Schneider'
        where a.NAME = 'Shnyder';
```

3.2. Aufbau

a) Datenablage

Die anfallenden dauerhaften Daten (Nutzdaten und organisatorisch notwendige Informationen) liegen in besonders strukturierter Form auf einem Massenspeicher (Festplatten).

Beispiel: Tabellen; Beziehungen



b) Speicherresidente Teile / SGA

   Ungespeicherte Werte, Cacheinhalt, Sortierungen, Abfragen usw. werden in einem fest dem RDBMS zugeordneten Speicherbereich (System Global Area SGA) vorgehalten.

   Beispiel: Ergebnismenge eines SELECT; Daten die „gerade eben“ schon einmal von der Festplatte gelesen wurden

c) Prozesse

   Die Bedienung von Benutzeranfragen sowie alle Formen von datenbankinternen Vorgängen werden durch Datenbank-Prozesse erledigt. Es handelt sich dabei um fallweise automatisch gestartete Programme, die ihre Aufgaben erledigen und dann wieder beendet werden, bzw. in einen Schlafzustand fallen.

   Beispiel: Ankommende Netzwerkverbindung; Zurückschreiben von Daten aus dem Speicher auf die Festplatten

4. Transaktionen

4.1. Prinzip

Eine Transaktion fasst in sich mehrere logisch zusammengehörige, jedoch nicht zwangsweise technisch voneinander abhängige Arbeitsschritte zusammen.

Beispiel: Überweisung von Bankkonto zu Bankkonto.

- Geld wird von Konto A abgebucht
- Das System möchte Geld auf Konto B gutschreiben, stürzt aber ab.
- Nach dem Wiederanlauf fehlt der Überweisungsbetrag, er wurde abgebucht, jedoch nie angewiesen.

4.2. Konsequenz: Absicherung

Diese Art Risiko sehen die Anwender gar nicht gerne, es muß also ein Schutzverfahren gefunden werden. Die Lösung ist der transaktionsorientierte Denkansatz: Alle logisch zusammengehörigen Aktivitäten müssen entweder glücken, oder rückgängig gemacht werden.

In unserem Beispiel fassen wir also den Abbuchungsvorgang von Konto A und die Gutschrift auf Konto B als zusammengehörig auf – funktioniert demzufolge einer der beiden Vorgänge nicht, muß es sein als hätte der jeweils andere ebenfalls nie stattgefunden. Es darf kein Storno notwendig werden!

4.3. Rollback und Commit

Ein modernes Datenbanksystem handelt auf diese Weise: Man öffnet (bei vielen RDBMS implizit) eine Transaktion, führt zwei UPDATE-Befehle aus und schließt die Transaktion mit dem Befehl COMMIT wieder. Wird die Transaktion vor einem (gewaltsamen) Stop des Datenbanksystems nicht abgeschlossen, so hat es sie nie gegeben.

Als sehr angenehmer Nebeneffekt eröffnet sich dem Benutzer die Möglichkeit, seine Transaktion aus eigenem Willen (also ohne Crash) wieder rückgängig zu machen: In unserem Beispiel nehmen wir an, Konto B sei ein Sparkonto mit Maximalbetrag, und dieses Maximum würde mit der Gutschrift überschritten. Hat man das (schlechter Stil!) nicht zuvor überprüft, kann man den ersten UPDATE (Verringerung des Saldo auf Konto A) mit dem Befehl ROLLBACK jederzeit rückgängig machen.

4.4. Absicherung / Undo

Die datenschreibenden Befehle von SQL (sog. DML - „Data Manipulation Language“) legen vor jeder Veränderung eine Sicherheitskopie der alten Daten an, und geben den durch sie neu dargestellten Stand der Daten erst nach einem COMMIT für die Allgemeinheit zum Lesen frei.

4.5. Technischer Ablauf

a) Abfrage
   - Auswerten des SQL-Befehls
   - Ermitteln der Menge betroffener Daten
   - Einlesen der Daten in den Cache
   - Formulierung der Ausgabe
   - Weiterleitung an den User

b) Update + Commit
   - Auswerten des SQL-Befehls
   - Ermitteln der Menge betroffener Daten
   - Einlesen der Daten in den Cache
   - Erstellen von Sicherheitskopien
   - Verändern der betroffenen Blöcke
   - Freigeben der geänderten Daten für Lesezugriffe

c) Theorie für einen Join

```
select b.AUFTRAG_NR
 from TAB_KUNDEN a, TAB_AUFTRAEGE b
 where a.KD_NR=b.KD_NR  and a.KD_NR=1122;
```

   - Möglichkeit 1 – nested loop („verschachtelte Schleife“)

     Für jede Zeile in Tabelle TAB_KUNDEN (a) wird Tabelle TAB_AUFTRAEGE (b) komplett durchsucht.

   - Möglichkeit 2 – hash join
     Vorab zur Erklärung: „hash“ ist hier ein mathematisches Verfahren zur Erzeugung einer deutlich kleineren, aber trotzdem eindeutigen ID.

     Stark vereinfacht: Die Schlüsselwerte der Tabelle mit der kleineren betroffenen Zeilenanzahl werden zuerst in einen hash-Wert umgewandelt. Die Schlüssel der größeren Menge werden zur Laufzeit in hash umgerechnet und dann mit dem (zuvor berechneten) hash-Schlüssel der kleineren Tabelle verglichen.

     Durch die deutlich kürzeren hash-Schlüssel ergibt sich in verschiedenen Situationen ein (Geschwindigkeits-)Vorteil, auch wenn das Verfahren im ersten Augenblick umständlicher erscheint.

5. Cache

5.1. Exkurs

a) Diverse Größenordnungen

	Größe	**Preis pro GB**	**mittlere Zugriffszeit**
RAM-Modul	2GB - 8GB	20 EUR - 150 EUR	0,000 000 1s (100ns)
Server-RAM	16GB - 256GB	dto.	dto.
Festplatte	146GB - 1TB	0,32 EUR - 0,24 EUR	0,008s - 0,012s (~10ms)
RAID-Array Kapazität	146GB – 10TB	1,44 EUR - 10 EUR (!!)	0,001s - 0,010s (~8ms)

b) Funktionsweise von Festplatten

- Magnetisierbare, sich sehr schnell drehende Scheibe
- Mechanisch positionierter Schreib- und Lesekopf
- Anfahren nebeneinanderliegender Positionen: **schnell**
- Anfahren verstreuter Positionen: **langsam**

5.2. Caching

a) Schreib-Cache

Vorschalten von RAM vor den Festplattenzugriff.

Die zu schreibenden Daten werden vor der „Übergabe“ an die Festplatte so sortiert, daß deren Schreibkopf möglichst wenige verschiedene Positionen anfahren muß.

Außerdem stehen Daten aus dem Schreibcache auch Leseprozessen analog zum Lese-Cache zur Verfügung.

b) Lese-Cache

Nachschalten von RAM nach dem Festplattenzugriff.

Das Verfahren das der Schreibcache verwendet ist hier nicht anwendbar, da nachträglich keine Einflußnahme auf die Speicherorte mehr möglich ist. Daher werden alle Daten, die (mit mehr oder weniger Kopfbewegungen) gelesen wurden, im RAM gehalten, bis dieser voll ist. Der Vorteil liegt nun darin, daß ein erneuter Lesezugriff ohne Inanspruchnahme der Festplatte erfolgen kann..

5.3. Risiko

a) Schreib-Cache

   RAM kann ohne Spannungsversorgung keine Informationen behalten. Das schreibende Programm geht aber davon aus, daß die Daten bereits auf den statischen Datenträgern sicher sind. Bei Stromausfall oder Rechnerabsturz ist das ungünstig! :(

b) Lese-Cache
   Kein Risiko, nur Vorteile.

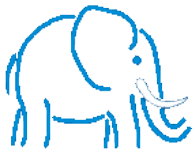
5.4. Absicherung / Redo

Die Schreibvorgänge in einen Cache hinein haben wie oben angeführt ihre Vorteile und Risiken. Ein Datenbank-Managementsystem geht hier zur Absicherung gegen Stromausfall / Absturz einen Mittelweg: Bei der Veränderung von Daten wird nicht der gesamte Datensatz auf die Platte geschrieben, sondern voll in den Cache. Auf die Platte wird nur die Differenz zwischen altem und neuem Stand sofort gesichert.

Vorteil: Kleine Datenmengen; unmittelbar aufeinander folgende Positionen auf der Festplatte -> Schnell.

6. Klassifikation Datenbanksysteme

- Tabellenkalkulation
  - Excel
  - OpenOffice Calc
- Workstation-Level
  - Access
  - BerkeleyDB
- Mittelklasse
  - MySQL
  - PostegreSQL
- Enterprise-Niveau
  - Oracle
  - IBM DB2
  - Microsoft SQL-Server
  - Informix
  - Sybase



7. Anwendung in der Wirtschaft

7.1. OLTP (= Online Transaction Processing, Echtzeitbetrieb)

a) Kunden-, Adress- und/oder Artikeldatenbank
b) Onlineshop, Internet- / Intranet-Backend (dynamische Webseiten)
c) Automatisiertes Lager

7.2. OLAP (= Online Analytical Processing, Data Warehouse / Data Mining)

a) Auswertungen zur Wirtschaftlichkeit, Controlling
b) Kundenprofil-Erstellung
c) u.U. Rasterfahndung